# 整数行列のFrobenius標準形のモジュラー計算法(II) (Computer Algebra : Algorithms, Implementations and Applications)

AUTHOR(S):

森継, 修一; 栗山, 和子

数理解析研究所講究録 1295 巻 2002 年 87-92



# 整数行列のFrobenius標準形のモジュラー計算法 (II) *

図書館情報大学　森継修一　(Shuichi Moritsugu) †
国立情報学研究所　栗山和子　(Kazuko Kuriyama) ‡

## 1　はじめに

一般に線形代数の教科書では、行列の代表的な標準形としてJordan標準形が示されていることが多いが、これを数式処理で計算すると固有値を代数的数として扱う必要があり、拡大体上の計算の効率化に難点がある。これに対しFrobenius標準形（有理標準形）は、体の拡大を必要とせず行列要素間の四則演算（有理演算）のみで計算が可能であり、その結果はブロックの並び順まで含めて一意的である。さらにFrobenius標準形は、もとの行列の特性多項式・最小多項式、固有値の代数的・幾何学的重複度や対応する（一般）固有ベクトルの構成などの完全な情報をJordan標準形と同等に含んでおり [9][12]、記号的線形代数計算法を考えるにはJordan標準形よりもFrobenius標準形を基礎とする方が適している。また、固有値法による連立代数方程式の解法 [14][10][11] への応用も提案されている。

筆者らは以前、Frobenius標準形の計算において、有理数計算を避けると同時に要素の成長を最大限抑える「分数なし計算法」[7] を発表した。本研究はここにモジュラー計算法の適用を試みるものであるが、[13] として報告した時点では、必ずしも十分な効果が得られていなかった。本稿においては、主として、変換行列の計算法を変更することによる計算の高速化の実験的評価について報告する。

## 2　行列のFrobenius標準形

以下の議論は任意の体の元を要素とする行列に対して成り立つが、具体的には、有理数を要素とする行列 $A=[a_{ij}]$, $a_{ij}\in\mathbf{Q}$ を考える。

**定義 1 (コンパニオン行列)**
次の $n\times n$ 正方行列

$$C=\begin{bmatrix} 0 & 1 & & & \\ 0 & 0 & \ddots & & O \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & 1 \\ c_0 & c_1 & \cdots & c_{n-2} & c_{n-1} \end{bmatrix} \tag{1}$$

は、多項式 $f(x)=x^n-c_{n-1}x^{n-1}-\cdots-c_1x-c_0$ に随伴するコンパニオン行列と呼ばれる。特に、1次多項式 $f(x)=x-c_0$ のコンパニオン行列は $1\times 1$ 行列 $[c_0]$ とする。

*本研究は「平成13年度図書館情報大学特別研究 (C)」の助成に基づく。
†moritsug@ulis.ac.jp
‡kuriyama@nii.ac.jp

行列 $C$ (1) の特性多項式 $\varphi_C(x)$ と最小多項式 $\phi_C(x)$ は、$f(x)$ に一致する。

**定理 3 (Frobenius 標準形)**
任意の $n \times n$ 正方行列 $A$ は、適当な正則行列 $S$ により次の形のブロック対角行列に一意的に相似変換され、これを $A$ の Frobenius（または有理）標準形という。

$$F = S^{-1}AS = C_1 \oplus C_2 \oplus \cdots \oplus C_t. \tag{2}$$

各ブロック行列 $C_i\,(i = 1, \cdots, t)$ は、$m_i \times m_i$ コンパニオン行列 (1) であり、$C_{i+1}$ の随伴多項式 $\varphi_{i+1}(x)$ は、$C_i$ の随伴多項式 $\varphi_i(x)$ を割り切る $(i = 1, \ldots, t-1)$。さらに $A$ の最小多項式は $\phi_A(x) = \varphi_1(x)$、特性多項式は $\varphi_A(x) = \varphi_1(x) \cdot \varphi_2(x) \cdots \varphi_t(x)$ で与えられる。

行列を (2) に類似のブロック対角形に変換して特性多項式を求める方法が Danilevskii 法 [2][3] として古くから知られているが、本稿では、韓・伊理 [6] によるアルゴリズムに従う。

**定義 4 (基本変形)**
任意の $n \times n$ 正方行列 $A$ に対する次の 3 つの操作を基本変形と呼ぶ。

$op1(k, \ell)$ ：$A$ の第 $k$ 行と第 $\ell$ 行を入れ換え、続いて第 $k$ 列と第 $\ell$ 列を入れ換える。
$op2(k, c)$ ：$A$ の第 $k$ 行を $c^{-1}$ 倍し、続いて第 $k$ 列を $c$ 倍する。
$op3(k, \ell, c)$ ：$A$ の第 $k$ 行に第 $\ell$ 行の $c$ 倍を加え、続いて第 $\ell$ 列から第 $k$ 列の $c$ 倍を引く。

基本変形を順次適用することは、消去に必要な相似変換 $\cdots S_3^{-1}\left(S_2^{-1}\left(S_1^{-1}AS_1\right)S_2\right)S_3 \cdots$ を計算することに相当し、最終的に式 (2) における $F, S, S^{-1}$ が得られる。

**注意 1**
コンパニオン行列・Frobenius 標準形として (1) (2) の転置で定義する流儀もある。その場合は、$F^T = S^{-1}A^TS \iff F = S^TAS^{-T}$ により、相互に変換可能である。

**注意 2**
行列 $A$ の要素をすべて整数にとった場合、特性多項式 $\varphi_A(x) = \det(xE - A)$ の係数が必ず整数になるため、Frobenius 標準形の各要素も整数となる。ただし、変換行列 $S, S^{-1}$ については、いずれか一方しか整数行列にとることはできない。

## 3 モジュラー計算アルゴリズム

以下では整数行列 $A$ のみを対象とし、固有ベクトルの計算への応用を想定して、$AS = SF$ をみたす $F, S$ を求めることとし、$S$ の各要素が整数になるようアルゴリズムを構成する。

基本変形 $op2(k, c)$ は「$c^{-1}$ 倍する」という除算を含むが、基本変形における有理演算はすべて素数 $p$ を法として実行可能（Euclid 互除法により、$s \equiv c^{-1} \pmod{p}$ が計算可能）なので、$\mathbf{Z}_p$ での Frobenius 標準形 $A_pS_p = S_pF_p$ が同じプログラムで求められる。複数の法による結果からの $\mathbf{Z}$ 上での $F, S$ の復元には、Chinese Remainder Theorem を利用する。

**定理 5 (CRT：法が 2 つの場合)**
$m_1, m_2$ が互いに素な整数のとき、連立合同式の解は、$m_1s + m_2t = 1$ を満たす 1 組の整数 $s, t$ を用いて、次の式で与えられる。

$$\begin{cases} x \equiv a_1 & (\bmod\ m_1) \\ x \equiv a_2 & (\bmod\ m_2) \end{cases} \quad \Longrightarrow \quad x \equiv a_1m_2t + a_2m_1s \quad (\bmod\ m_1m_2)$$

したがって、素数 $p_1$, $p_2$, $p_3$,... に対して、法を $p_1$, $p_1p_2$, $p_1p_2p_3$,... と上げていくには、

$$\begin{cases} x \equiv a_{k-1} & (\mathrm{mod}\ p_1 \cdots p_{k-1}) \\ x \equiv a_k & (\mathrm{mod}\ p_k) \end{cases} \qquad (k = 2,\ 3, \ldots)$$

に対して、定理 5 を繰り返し適用する（Newton の補間公式）。一般に、変換行列 $S$ の要素の成長が著しいので、mod $p_1 \cdots p_{k-1}$ での値 $S^{(k-1)}$ と mod $p_1 \cdots p_{k-1}p_k$ での値 $S^{(k)}$ とが一致したら、整数上で $AS^{(k)} = S^{(k)}F^{(k)}$ を調べて終了判定とする。

**アルゴリズム 1 (Frobenius 標準形のモジュラー計算)**

% 入力：整数要素の $n \times n$ 行列 $A$　　ある程度大きな素数のリスト $\{p_1, p_2, \ldots, p_s\}$
% 　計算過程での $F^{(k)}, S^{(k)}$ は mod $p_1 \cdots p_k$ での値を表す。
% 出力：$A$ の Frobenius 標準形 $F$ と相似変換行列 $S$　（$AS = SF$）
Compute $F_{p_1}, S_{p_1}$ s.t. $A_{p_1}S_{p_1} \equiv S_{p_1}F_{p_1}$ (mod $p_1$);
$k := 1$;　$F^{(1)} := F_{p_1}$;　$S^{(1)} := S_{p_1}$;
loop : Do until $(S^{(k-1)} = S^{(k)})$
　　$k := k + 1$;
　　Compute $F_{p_k}, S_{p_k}$ s.t. $A_{p_k}S_{p_k} \equiv S_{p_k}F_{p_k}$ (mod $p_k$);
　　Construct $F^{(k)}$ from $F^{(k-1)}$ and $F_{p_k}$ by CRT;
　　Construct $S^{(k)}$ from $S^{(k-1)}$ and $S_{p_k}$ by CRT;
　If $(AS^{(k)} = S^{(k)}F^{(k)}$ (over $\mathbf{Z}))$ then return $\{F^{(k)}, S^{(k)}\}$ else goto loop;

## 4 unlucky な素数の発見と回避

素数 $p$ が unlucky な場合、すなわち、$\mathbf{Z}$ 上で求めた標準形 $AS = SF$ と $\mathbf{Z}_p$ での標準形 $A_pS_p = S_pF_p$ とにおいて、$F \not\equiv F_p \pmod p$ または $S \not\equiv S_p \pmod p$ となる場合が起こりうる。このとき、1 組でも unlucky な $F_p, S_p$ が含まれていると、いくら法 $p_i$ の個数を増やしても CRT によって $F, S$ を復元することができなくなる。この識別は、次による。

**補題 6 (Howell[5])**
lucky な素数 $p$ を法とする場合の pivot 選択のパターンは、$\mathbf{Z}$ 上での計算で起きる pivot 選択のパターンと一致する。

消去計算の過程における pivot 選択の履歴を記録しておき、unlucky な素数 $p$ が偶然 pivot 要素を割り切った場合の、本来起きないはずの行交換・列交換 $op1(k, \ell)$ を識別し、その mod $p$ での計算結果は捨てる。各ステップにおける pivot の素因数は有限個なので、特定の行列に対する unlucky な素数は全体として有限個しか存在しないことになる。

ただし、$\mathbf{Z}$ 上での計算過程における「正しい pivot 選択のパターン」は未知なので、現在の実装では、最初の 3 つの素数 $p_1, p_2, p_3$ でのパターンで多数決を行ない、その結果を正しいものと仮定して計算を進めている。万一、この仮定が間違っていた場合は、その後に unlucky な素数が頻出することになるので、最初に戻って「正しいパターン」の推定からやり直すこととする。この点で、確率的なアルゴリズムであることは避けられないが、実用上は、できるだけ大きめの素数を法にとることにより、失敗の確率はかなり低く抑えられる。実験例では、テストのため意図的に 2〜3 桁の素数を用いた場合には unlucky な素数が頻出したが、10 桁程度の一連の素数を法にとった場合には、すべて lucky であった。

## 5 変換行列の構成法

アルゴリズム1では、相似変換のための列操作を順次適用していく ($S = S_1 S_2 \cdots$) と、一般に変換行列 $S$ の要素は巨大な整数に成長してしまう。相似変換行列 $S$ は一意ではないため、$AS = SF$ をみたす正則なものが1つ見つかれば十分である。したがって、Frobenius 標準形 $F$ だけを先に求め、その後で、$AS = SF$ をみたす正則行列 $S$ (しかもできるだけ簡潔な要素を持つもの) を構成することを考える。$F$ の要素は $S$ ほどには成長しないので、モジュラー法を適用する場合、法 $p_i$ の個数がごく少数ですむことが期待できる。

次の関係式において、$S = [s_1 \mid s_2 \mid \cdots \mid s_n]$ として両辺を列毎に比較すると

$$AS = SF = S \begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix} \quad \Longrightarrow \quad \begin{cases} As_1 &= c_0 s_n \\ As_2 &= s_1 + c_1 s_n \\ & \cdots \\ As_n &= s_{n-1} + c_{n-1} s_n \end{cases}$$

となる。下の行から順に $s_{n-1}, \ldots, s_1$ を消去すると次を得る。

$$\left(A^n - c_{n-1}A^{n-1} - \cdots - c_1 A - c_0 E\right) s_n = \mathbf{0} \qquad \text{すなわち} \qquad \varphi(A) s_n = \mathbf{0} \tag{3}$$

- 最初のコンパニオンブロックに対しては、$\varphi(x)$ は最小多項式なので、Cayley-Hamilton の定理により $\varphi(A) = O$ が成り立つから、$s_n$ は任意のベクトルにとることができる。
- 2番目以降のブロックに対しては、斉次方程式 (3) を実際に解いて $s_n$ を求める。

残りの列ベクトルは、漸化式 $s_j = As_{j+1} - c_j s_n \quad (j = n-1, \ldots, 1)$ より求める。

**注意 3**

この算法で得られる $S$ は、$AS = SF$ をみたしているが、その正則性については保証がない。(3) の解として特殊なベクトル $s_n$ を採らない限り $S$ は正則になると考えられるため、現在の実装では、解に含まれる任意定数に $\{+1, -1\}$ をランダムに割り振っている。この点で確率的であるが、得られた $S$ の正則性は、rank を求めて確認している。

**アルゴリズム2 (Frobenius 標準形のモジュラー計算＋変換行列の再構成)**

% 入力：整数要素の $n \times n$ 行列 $A$　　ある程度大きな素数のリスト $\{p_1, p_2, \ldots, p_s\}$
% 　　計算過程での $F^{(k)}$ は mod $p_1 \cdots p_k$ での値を表す。$S_{p_k}$ は保存しない。
% 出力：$A$ の Frobenius 標準形 $F$ と相似変換行列 $S$　　($AS = SF$)
Compute $F_{p_1}$ s.t. $A_{p_1} S_{p_1} \equiv S_{p_1} F_{p_1} \pmod{p_1}$;
$k := 1$;　　$F^{(1)} := F_{p_1}$;
loop : Do until $(F^{(k-1)} = F^{(k)})$
　　$k := k + 1$;
　　Compute $F_{p_k}$ s.t. $A_{p_k} S_{p_k} \equiv S_{p_k} F_{p_k} \pmod{p_k}$;
　　Construct $F^{(k)}$ from $F^{(k-1)}$ and $F_{p_k}$ by CRT;
If $\varphi(A) \neq O$ then goto loop;　　% $\varphi(x)$ = min.pol. of $F^{(k)}$
Construct $S$ from $A, F^{(k)}$　(over $\mathbf{Z}$);
While rank$(S) < n$ (over $\mathbf{Z}$) do reconstruct $S$;
return $\{F^{(k)}, S\}$

表 1: CPU-Time(sec) for integer matrices I

| | Algorithm-1 | | | Algorithm-2 | | | | Maple | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\#p_i$ | len$\|s_{ij}\|$ | $T_1$ | $\#p_i$ | len$\|s_{ij}\|$ | $T_2$ | $T_2/T_1$ | $T_M$ | $T_2/T_M$ |
| 12 | 31 | 279 | 16.23 | 7 | 47 | 2.05 | 0.126 | 13.20 | 0.155 |
| 14 | 43 | 384 | 44.02 | 8 | 55 | 4.25 | 0.097 | 39.17 | 0.109 |
| 16 | 56 | 507 | 103.42 | 9 | 63 | 8.30 | 0.080 | 100.53 | 0.083 |
| 18 | 71 | 651 | 227.38 | 10 | 73 | 16.09 | 0.071 | 229.86 | 0.070 |
| 20 | 89 | 818 | 485.72 | 11 | 81 | 29.28 | 0.060 | 497.32 | 0.059 |
| 25 | 141 | 1301 | 2391.33 | 13 | 103 | 112.31 | 0.047 | 2496.91 | 0.045 |
| 30 | 206 | 1911 | 9333.34 | 15 | 126 | 361.43 | 0.039 | 9003.52 | 0.040 |

表 2: CPU-Time(sec) for integer matrices II

| | Algorithm-1 | | | Algorithm-2 | | | | Maple | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\#p_i$ | len$\|s_{ij}\|$ | $T_1$ | $\#p_i$ | len$\|s_{ij}\|$ | $T_2$ | $T_2/T_1$ | $T_M$ | $T_2/T_M$ |
| 12 | 10 | 75 | 3.06 | 3 | 25 | 1.22 | 0.399 | 2.47 | 0.494 |
| 14 | 15 | 123 | 7.83 | 3 | 38 | 2.11 | 0.269 | 6.21 | 0.340 |
| 16 | 16 | 133 | 11.46 | 3 | 36 | 3.73 | 0.326 | 13.38 | 0.279 |
| 18 | 18 | 157 | 18.84 | 3 | 43 | 6.10 | 0.324 | 27.88 | 0.219 |
| 20 | 21 | 181 | 31.09 | 3 | 44 | 9.82 | 0.316 | 33.29 | 0.295 |
| 25 | 32 | 288 | 123.80 | 3 | 78 | 40.16 | 0.324 | 175.13 | 0.229 |
| 30 | 45 | 406 | 366.10 | 4 | 102 | 124.98 | 0.341 | 427.50 | 0.292 |
| 42 | 93 | 851 | 3546.36 | 4 | 146 | 997.12 | 0.281 | 1544.37 | 0.646 |

## 6 実験的評価

環境　H9000VK270 (CPU : PA-8000, 160MHz) メモリ 128MB 使用
HP-UX 版 Reduce3.6[4] + RLISP '88[8]

素数リスト $\{p_1, p_2, \ldots, p_{1000}\}$ = {2147483647, 2147483629, ..., 2147462143}

表 1 の行列は、$-10000 \sim 10000$ の範囲のランダムな整数を要素としたもので、コンパニオンブロック 1 つの標準形となるため、変換行列 $S$ を構成する際に斉次方程式 (3) を解く必要がない。その結果、アルゴリズム 2 では計算時間が大幅に改善されている。

表 2 の行列は、ランダムな係数をもつ多項式に随伴するコンパニオンブロックから、$12(8,4) \sim 42(12,10,8,6,4,2)$ という構造を持つように作成した。変換行列 $S$ を構成する際に斉次方程式 (3) を実際に解いているため、表 1 ほどではないが、速度は向上している。

他のシステムでは、Maple[1] のみが Frobenius 標準形を計算する組込関数（アルゴリズムは不明）を持つが、同じテスト行列に対して、計算速度はアルゴリズム 2 に及ばない。

## 7 まとめと今後の課題

(1) unlucky な素数の発見・回避についてはほぼ実現し、失敗の確率はかなり低く抑えられたと考えられる。

(2) 変換行列 $S$ を $A, F$ から再構成するアルゴリズムは、計算速度向上の点で有望である。実験では非正則な $S$ が生成されることはなかったが、確率的部分の評価が課題である。

(3) Maple の組込関数 frobenius については、変換行列として $\left(S^T\right)^{-1}$ を求める仕様になっているので、本研究と同じ用途に適用するためには、さらに逆行列の計算を要する。

(4) 有理数要素の行列へ拡張するためには、モジュラー計算からの有理数の復元はかなり計算が重いので、どの段階で $\mathbf{Z}_p$ 上から $\mathbf{Q}$ 上へ戻すのがよいか、比較検討を要する。

## 参 考 文 献


[1] Char, B. W., et al.: *Maple V Library Reference Manual*, Springer, N.Y., 1991.

[2] Danilevskii, A. M.: On the numerical solution of the secular equation, *Mat.Sb.*, **2**(1), 1937, 169–172. (in Russian).

[3] ファジェーエフ, ファジェーエバ: 線型代数の計算法 (上), 産業図書, 東京, 1970.

[4] Hearn, A. C.: *Reduce User's Manual (Ver. 3.6)*, RAND Corp., Santa Monica, 1995.

[5] Howell, J. A.: An Algorithm for the Exact Reduction of a Matrix to Frobenius Form Using Modular Arithmetic. I & II, *Math.Comp.*, **27**(124), 1973, 887–920.

[6] 韓太舜, 伊理正夫: ジョルダン標準形, 東京大学出版会, 東京, 1982.

[7] 栗山和子, 森継修一: 行列の有理標準形の分数なし計算法, 日本応用数理学会論文誌, **6**(3), 1996, 253–264.

[8] Marti, J.: *RLISP '88: An Evolutionary Approach to Program Design and Reuse*, World Scientific, Singapore, 1993.

[9] 森継修一, 栗山和子: 行列の Jordan 標準形の数式処理による厳密計算法, 日本応用数理学会論文誌, **2**(1), 1992, 91–103.

[10] Moritsugu, S. and Kuriyama, K.: On Multiple Zeros of Systems of Algebraic Equations, *ISSAC 99*, ACM, N.Y., 1999, 23–30.

[11] Moritsugu, S. and Kuriyama, K.: A Linear Algebra Method for Solving Systems of Algebraic Equations, *J.JSSAC*(日本数式処理学会誌), **7**(4), 2000, 2–22.

[12] 森継修一, 栗山和子: 行列の固有値・固有ベクトル・一般固有ベクトルの数式処理による記号的計算法, 日本応用数理学会論文誌, **11**(2), 2001, 103–120.

[13] 森継修一, 栗山和子: 整数行列の Frobenius 標準形のモジュラー計算法, 京都大学数理解析研究所講究録, **1199**, 2001, 220–227.

[14] 竹島卓, 横山和弘: 連立代数方程式の一解法 - 剰余環上の線形写像の固有ベクトルの利用, 数式処理通信, **6**(4), 1990, 27–36.